VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practise in the Company

2010 Adam Glumbík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Adam Glumbík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Absolvování individuální odborné praxe
Individual Professional Practise in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Městská nemocnice Ostrava, příspěvková organizace
2. Struktura závěrečné zprávy:
   a. Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b. Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c. Zvolený postup řešení zadaných úkolů.
   d. Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e. Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Dr.Ing. Eduard Sojka**

Konzultant bakalářské práce: Ing. Vasil Waliszewski

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

„Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

4.5. 2010 Adam Glumbík

Děkuji touto cestou p. doc. Dr.Ing. Eduardu Sojkovi a p. Ing. Vasilu Waliszevskemu za odborné vedení práce, podnětné rady a připomínky, které mi napomohly k sepsání této bakalářské práce.

Abstrakt

Bakalářská práce pojednává o mé absolvované praxi v Městské nemocnici v Ostravě. Uvádím v ní své výsledky, kterých jsem docílil v průběhu trvání praxe. Mým hlavním úkolem bylo podílet se na vytváření nové verze aplikace používané na oddělení sternální evidence. Aplikace byla vytvářena ve dvoučlenném týmu pod dohledem vedoucího odboru informatiky Městské nemocnice v Ostravě Ing. Vasilem Waliszevskim. Před začátkem programování jsme provedli rozdělení jednotlivých prací na aplikaci. Jednotlivé součásti aplikace byly vždy před přidáním do výsledného programu konzultovány a implementovány za přítomnosti druhého člena týmu. Tím jsme docílili, že oba členové týmu jsou seznámeni s celým zdrojovým kódem aplikace a jsou schopni v případě potřeby provádět zásahy i do části kódu, kterou vytvářel druhý člen. Přínosem pro mě bylo vyzkoušení práce v týmu, která je v dnešní době vyžadována téměř v každé firmě zabývající se vytvářením softwaru a dále zdokonalení znalostí programování v jazyce C#.

Abstract

Bachelor thesis is about my absolved professional practise in Municipal Hospital in Ostrava. I mention in it my results, which I have reached during the time of professional practise. My main task was to participate in creating new version of application used for sternal separation records. Application was developed in two-person team under the supervision of the Information Technology Department of the Municipal Hospital in Ostrava, Ing. Vasilem Waliszevskim. Before we start programming, we have divided work on application. The individual components of the applications were always before adding it to the final program, consulted and implemented in the presence of a second member of the team. Thus we have achieved that both team members are acquainted with the entire source code of application and are able, if necessary, intervening into the code creating by a second member of the team. The benefit for me was to try working in a team, which today is required in almost every company involved in developing software and improving my knowledge of programming in C#.

Klíčová slova

.NET, C#, Microsoft Visual Studio, Microsoft SQL Server, Windows Forms, Microsoft, Uživatelsky přívětivé rozhraní, Sternální evidence, Cytologie kostní dřeně

Key words

NET, C #, Microsoft Visual Studio, Microsoft SQL Server, Windows Forms, Microsoft, User-friendly interface, Sternal records, Bone marrow cytology

Obsah

1. Úvod

Překvapilo mě, kolik čerstvých absolventů vysoké školy elektrotechnické, má v dnešní době po ukončení studia problémy se získáním práce. Nikdo samozřejmě nemůže počítat s tím, že jen vysokoškolský titul mu zajistí vysněnou kariéru. Co tedy současné firmy nejvíce vyžadují? Většina nabídek práce uvádí požadavky jako je například praxe v oboru, samostatný a komunikativní přístup, schopnost práce v týmu, samostatné rozhodování a odpovědný přístup k zadaným úkolům.

Jedná se jen o stručný výčet schopností, které firmy od uchazečů vyžadují. Tím nejčastějším důvodem neúspěchu při ucházení se o práci, je u absolventů většinou malá nebo žádná praxe v oboru.

Proto jsem byl rád, že mi bylo v rámci bakalářské práce umožněno, vyzkoušet si, jak skutečná práce programátora vypadá.

Má bakalářská práce je obsahově rozdělena na 3 větší celky, z nichž každý popisuje jednu z fází vykonané praxe.

Před začátkem programování bylo potřeba udělat přesnou specifikaci všech funkcí, které bude výsledná aplikace poskytovat. Tato fáze velmi pomohla při úvodním rozdělení úkolů mezi jednotlivé členy týmu a zároveň odhalila některé z možných problémů jak při implementaci aplikace, tak při jejím nasazení do ostrého provozu.

V druhé části se zabývám samotnou implementací těch částí aplikací, které jsem měl za úkol vypracovat já. Vždy se snažím detailně popsat, proč používám zvolený postup a také ho předvádím na uvedeném zdrojovém kódu.

Poslední část se zabývá závěrečným zhodnocením bakalářské praxe s nasazením aplikace do ostrého provozu v prostředí nemocnice, čímž jsme docílili, že naše práce nezůstane pouze jedním z mnoha vyvíjených školních projektů, které nikdy nespatří nikdo jiný kromě nás a vyučujících.

2. Údaje o zadávající firmě

2.1 Městská nemocnice Ostrava

Pro svou praxi jsem si vybral Městskou nemocnici v Ostravě. Velmi mě vždy zajímalo, jak lze v dnešní době využít informační technologie ve spojení s lékařskou činností. První setkání s vedoucím odboru informatiky Ing. Vasilem Waliszevskim jsem absolvoval v květnu roku 2009. Nejdříve mě stručně seznámil s činností informatického oddělení v prostředí nemocnice. Mezi jejich hlavní činnost patří především správa všech počítačů v nemocnici. Vzhledem k počtu pracovních stanic, které se nacházejí jen na každém z oddělení nemocnice, mi bylo jasné, o jak časově náročnou práci se jedná. Dalším odvětvím činnosti oddělení je vývoj aplikací a právě do této oblasti jsem měl být přiřazen i já.

2.2 Pracovní zařazení studenta

Byl jsem společně s druhým studentem Kamilou Nykodýmovou, ucházející se také ve stejné době o praxi přiřazen do týmu, který měl na základě požadavků a původní verzi programu, vyvinout novou verzi aplikace pro oddělení klinické hematologie (OKH). Toto oddělení se dělí na 2 části. Klinická část - zabývající se diagnostikou primárních krevních onemocnění a hematologickou laboratoř kde se provádí všechna standardní hematologická vyšetření. Naše aplikace je určena právě pro hematologickou laboratoř, přesněji pro vedení sternální evidence. Abychom lépe porozuměli, k čemu přesně bude aplikace sloužit, bylo nám popsáno, jak takové vyšetření v laboratoři probíhá. Nejdříve se odebere kostní dřeň z hrudní kosti (sternum). Ta se následně nanese na sklíčko a poté se opticky, mikroskopem počítá a určí 1000 buněk v ní obsažených. Podle poměrů jednotlivých napočítaných buněk se následně stanoví diagnóza. A právě záznamy o počtu buněk se budou vkládat do naší aplikace.

Protože jsme měli možnost prohlédnout si původní aplikaci, udělali jsme si konkrétní představu o tom, jak by měla celá aplikace vypadat a také jaké funkce poskytovat. Na Obr.1, je vidět, jak vypadalo uživatelské rozhraní původní aplikace.

Obr.1 Původní vzhled aplikace při editaci karty pacienta

3. Zadané úkoly vypracovávané v průběhu praxe a jejich časová náročnost

3.1 Nový vzhled aplikace

Naším prvním úkolem bylo navržení nového, modernějšího vzhledu aplikace. Původní verze naprogramovaná v jazyce Pascal se ovládala pouze pomocí klávesnice a nenabízela možnost rychlé navigace mezi jednotlivými funkcemi programu. Mnohem pohodlnější je v dnešní době použití myši. Časová náročnost tohoto úkolu byla odhadnuta na jeden týden práce, po kterém jsme měli předložit,

jak by podle nás měla aplikace vypadat, kde se budou nacházet navigační prvky a jaké budou minimální požadavky na monitor ke správnému zobrazení aplikace.

Při vytváření vzhledu jsme vycházeli z rozložení prvků používaného v dnešní době především na internetových stránkách, kde se ovládací menu nachází na obrazovce stále na stejném místě, a to nejčastěji vlevo nahoře. Barvy a obrázky jsme použili z internetových stránek samotné nemocnice.

3.2 Implementace současných funkcí aplikace

Druhým úkolem byla implementace většiny stávajících funkcí původní verze aplikace. Tyto funkce se dělily na dvě skupiny. První skupina, která se bude muset implementovat ve stejné míře, jako byla využívána doteď a druhá skupina, jejíž funkčnost bude z různých důvodů poupravena nebo zcela zrušena. Implementování tohoto úkolu bylo stanoveno na dobu maximálně tří měsíců.

3.2.1 Funkce, které budou zachovány v plné míře

Funkce, které se i po čase ukážou jako dostačující k plné spokojenosti uživatele, není třeba měnit ani poupravovat. Pro nás to znamenalo podrobně analyzovat tyto funkce a implementovat je přesně tak**,** jak byly v původní aplikaci. Jediným rozdílem bylo jejich přizpůsobení pro naše ovládací prvky použité v aplikaci.

3.2.1.1 Správa záznamů jednotlivých pacientů

Hlavní funkcí, která tvoří základ pro celou aplikaci je správa záznamů jednotlivých pacientů. Pod pojmem správa je myšleno vytváření, editace, mazání a vyhledávání. K vytváření se používá speciální formulář, do kterého se kromě základních údajů jako jsou jméno, příjmení, rodné číslo, udávají také počty jednotlivých buněk v kostní dřeni. Do formuláře jsme implementovali, na některé ovládací prvky kontroly, ověřující správnost jejich vyplnění. O těchto kontrolách podrobně píši dále.

3.2.1.2 Kontroly při vytváření nového záznamu o pacientovi

Byli jsme informováni, že v původní aplikaci se kontroly dělily na dva typy. Na tvrdé, které při porušení definovaných pravidel znemožní uložení celého záznamu do databáze a na měkké kontroly, které vyvolávají hlášení o možnosti výskytu chybného vyplnění, ale samotnému uložení do databáze nezabrání.

Pod tvrdé kontroly spadají pole, jejichž obsah musí být vždy vyplněn pro jednoznačnou identifikaci pacienta. Takovými poli jsou například rodné číslo, příjmení a Myel. číslo. Pod měkké kontroly patří pak správnost vyplnění rodného čísla podle požadavků určených pojišťovnami a správnost součtu jednotlivých buněk v kostní dřeni.

3.2.1.2.1 Kontrola správnosti vyplnění rodného čísla

Rodné číslo je jednoznačný číselný identifikátor, který je přidělován obyvatelům České republiky. Lze z něj vyčíst datum narození a pohlaví příslušné osoby. Prvních šest číslic popisuje datum narození ve formátu RRMMDD (např. 880824 označuje muže narozeného 24. srpna 1988), přičemž ženám se k číslici měsíce připočítává 50 (885824 označuje ženu narozenou 24. srpna 1988). Od roku 2004 byla zavedena možnost, kdy v případě, že jsou vyčerpána všechna platná čtyřčíslí, použít alternativní rodné číslo, u kterého se mužům k číslu měsíce přičítává 20 a ženám 70. Zbytek rodného čísla odlišuje lidi narozené ve stejný den a zpravidla se odděluje lomítkem. Do 1. Ledna 1954 byly za lomítkem jen tři cifry, poté přibyla čtvrtá číslice, sloužící jako kontrolní číslice. Celé rodné číslo je pak beze zbytku dělitelné jedenácti. Jelikož je rok narození v rodném čísle uveden pouze dvoumístně, rodné číslo přestane být roku 2054 jednoznačné.

Informace o rodném čísle byla čerpána z: http://cs.wikipedia.org/wiki/Rodné_číslo, 20.2.2010, 19:30

Našim konzultantem jsme byli upozorněni i na možnost výskytu případů, kdy rodné číslo neodpovídá předepsanému formátu, a proto jsme kvůli zachování funkčnosti pro všechna možná rodná čísla (rodná čísla přistěhovalců apod.) implementovali tuto kontrolu jako měkkou.

3.2.1.2.3 Kontrola správnosti součtu jednotlivých buněk

Program má předně sloužit pro uchovávání informací o buňkách v kostní dřeni pacienta. Postup získání těchto informací jsem uvedl výše. Do aplikace se zadávají přesné počty buněk, které musí v konečném součtu dávat množství uvedené u každé konkrétní skupiny. Součet všech tří skupin nakonec dává dohromady 1000. Aby bylo umožněno dát si například při počítání buněk pauzu, jsou všechny kontroly tohoto typu implementovány jako měkké.

3.2.1.3 Tisk záznamů pacientů

Aplikace musí umožnit uživateli tisk jednotlivých záznamů o pacientech na všech dostupných tiskárnách. Vytištěná stránka musí mít přehlednou formu a měla by obsahovat všechny vyplněné údaje. Vzhled vytištěné stránky v nové aplikaci bude stejný, jako byl ve staré.

3.2.2 Funkce, které aplikace poskytovala, ale dále už nejsou potřeba

Původní aplikace umožňovala k záznamům, vložení několika typů komentářů. Po domluvě jsme se rozhodli všechny typy sjednotit a udělat z nich jeden komentář zahrnující v sobě všechny původní komentáře.

Také funkce zjištění podobnosti dvou rodných čísel pro stejně nebo podobně se jmenující osoby byla určena jako nadbytečná a tedy v naší verzi aplikace není zahrnuta. Původně tato funkce pomáhala v nemocnici zajišťovat ještě další kontrolu vyplněných údajů o pacientovi. Na druhou stranu nadbytek doplňkových kontrol může zhoršit celkovou snadnou použitelnost aplikace.

3.3 Nové funkce, které bude třeba navrhnout a implementovat

Nová aplikace přináší možnost využití nových moderních technologií a postupů, jak pro zlepšení původních funkcí, tak i pro implementování úplně nových, jejichž vytvoření by v předešlé verzi programu bylo velice obtížné. Také lze opravit chyby, na které se při provozování původní aplikace přišlo až po jejím finálním nasazení a náklady a složitost, které by vyžadovalo jejich řešení, je odsunulo až do další verze aplikace.

3.3.1 Návrh databáze a využití Microsoft SQL Serveru 2005

V původní verzi byly záznamy o pacientech ukládány do textových souborů. Z toho plyne, že jakékoliv složitější operace s daty se stávaly časově velmi náročné. Proto bylo jasné, že nová aplikace by měla k ukládání dat používat lepší mechanismus. Tímto mechanismem se pro nás stalo využití databázového serveru. Vzhledem k tomu, že se Městská Nemocnice v Ostravě zaměřuje převážně na produkty firmy Microsoft, bylo jasnou volbou použití Microsoft SQL Serveru. Návrh struktury databáze je popsán v části 5.1.

3.3.2 Správa použitých číselníků

Číselníky a jejich správa se stala jednou z nových funkcí aplikace, které musely být implementovány úplně od základů. Původní verze jejich správu neumožňovala, takže data v nich byla ze začátku staticky uložena. Číselníky používané v aplikaci se přitom poměrně často mění. Například ministerstvo zdravotnictví co čtvrt roku vydává aktualizované tabulky obsahující informace o zrušených, nahrazených a nově vzniklých pojišťovnách.

Naše aplikace pracuje s číselníky pojišťoven, diagnóz a oddělení. Uživatelé by si měli být schopni sami, pokud je například některá pojišťovna zrušena, popřípadě nahrazena jinou, tuto změnu provést sami přímo v rámci aplikace, bez nutnosti zásahu programátora.

3.3.3 Filtrování pacientů podle zadaných atributů

Užitečnou funkcí, která bude výrazně zrychlovat práci uživatelů s aplikací je filtrace záznamů. Setkáváme se s ní dnes všude. Například v internetových obchodech zadáme konkrétní parametry, které by měl námi požadovaný výrobek mít a server nám vrátí pouze ty záznamy, které odpovídají všem našim požadavkům. Pacienti se v naší aplikace filtrují podle svého příjmení, rodného čísla a Myel. čísla.

3.3.4 Přidávání obrázkových příloh k jednotlivým záznamům

Jednou z posledních funkcí na které jsme se domluvili s Ing.Vasilem Waliszevskim, a zároveň jednou z nejsložitějších na implementaci, bylo přidávání obrázkových příloh k jednotlivým záznamům pacientů. Jedná se o snímky pořízené laboratorním mikroskopem ve formátu JPG. Maximum obrázků, které bude možné vložit k jednomu záznam u pacienta je 5.

4. Rozdělení práce mezi jednotlivé členy týmu

Po definování všech požadavků na aplikaci i na použité technologie, zbývalo před jejich samotným vypracováním, už jen jejich rozdělení. Tím by měla být doba vypracování kratší, než kdyby každý dělal to, co by se mu v danou chvíli nejvíce zamlouvalo.

Podle svých schopností jsme se nakonec dohodli na následujícím rozdělení. Mými úkoly bylo propojení aplikace s databází, což v sobě zahrnuje i její návrh a realizaci v SQL Serveru a vytvoření create scriptů pro její realizaci na serveru v nemocnici. Dalším úkolem bylo vytvořit a realizovat vše potřebné pro tisk záznamů pacientů a jako poslední vymyslet a implementovat mechanismus, k přidávání obrázkových příloh u pacientů.

Hlavním úkolem druhého studenta bylo vymyslet vzhled celé aplikace, vytvořit administrační část, ve které si uživatelé mohou sami měnit obsah číselníků, popřípadě definovat cestu ke složce pro ukládání snímků a nakonec implementovat všechny kontroly při vyplňování nového záznamu o pacientovi.

Toto jednoduché rozdělení, nám umožnilo věnovat se konkrétním aspektům aplikace. Na některých částech nebylo možné začít pracovat dříve, než byly dokončeny jiné, takže jsme kromě rozdělení práce měli i přidělen čas, ve kterém jsme museli mít danou část aplikace hotovou alespoň natolik, aby mohl druhý člen bez problémů pokračovat ve své práci.

5. Postup při řešení jednotlivých úkolů

5.1 Návrh databáze a její realizace v prostředí Microsoft SQL Server 2005

5.1.1 Návrh a realizace databáze

Základem pro můj návrh databáze se staly atributy vyplňované při vytváření nového záznamu o pacientovi. Výsledný model databáze se nachází v příloze A. Je z něho vidět, že celá aplikace stojí na dvou tabulkách, z nichž jedna obsahuje konfigurační informace celé databáze a druhá samotné záznamy pacientů. Na tabulku s pacienty jsou napojeny tři číselníky. Číselník pojišťoven, diagnóz a oddělení.

Pro testovací účely jsem na svém počítači při vytváření aplikace využil Microsoft SQL Server 2005 Express, který je součástí instalačního balíčku Microsoft Visual Studia 2008. Součástí této instalace jsou i některé z nástrojů ke správě databáze. Vzhledem ke zkušenostem z předmětu Správa databází jsem ke správě použil SQL Server Management Studio.

Prostředí tohoto nástroje je přehledné a není problém se v něm během chvíle naučit orientovat. Kvůli optimalizaci výkonu jsem se zaměřil na časově nejnáročnější dotazy používané v databázi a pokusil se snížit jejich náročnost, pomocí indexů přiřazených na nejčastěji vyhledávané atributy.

V době, kdy jsem vyvíjel aplikaci, vyšla nová verze Visual Studia, Microsoft Visual Studio 2010 RC2. V rámci testování jsem ji nainstaloval a zjistil, že původní SQL Server 2005 Express byl nahrazen SQL Serverem 2008 Express. To samo o sobě není důležité, protože tabulky zůstaly beze změny zachovány, ale problém byl, že přestala fungovat Express verze SQL Server Management Studia, podporující jen verzi SQL Serveru z roku 2005.

Tímto jsem byl donucen hledat jiné nástroje, kterými bych mohl nadále spravovat vytvořenou databázi. V základní instalaci Visual Studia je už obsažen jen SQL Plus pracující pod příkazovým řádkem a vyžadujícím velmi dobrou znalost příkazů pro správu SQL Serveru. Proto jsem raději využil trial verze některých z komerčních nástrojů. Prvním bylo Aqua Data Studio sestavení 8.0.9 společnosti Aqua Fold. Jeho uživatelské rozhraní je velice podobné SQL Server Management Studiu. Ještě pro porovnání jsem se také na správu pokusil použít MS SQL Maestro společnosti SQL Maestro Group, které podle mě nabízí ještě lepší funkce než předešlé nástroje, samozřejmě ale také za odpovídající cenu.

U Aqua Data Studia je možné po vypršení testovací lhůty vyžádat novou bezplatnou 14-ti denní licenci, u MS SQL Maestro po 30 dnech testovací lhůta vyprší a pro její další použití by bylo nutné zakoupit plnou licenci.

5.1.2 Napojení ovládacích prvků aplikace na databázi

Když byla databáze realizovaná a nakonfigurovaná, zbývalo s ní ještě propojit aplikaci. Ve Visual Studiu jsem vytvořil pro spojení s databází nový Data Source. Ze seznamu jsem vybral databázi ke které se bude aplikace připojovat, čímž se automaticky vygeneroval připojovací řetězec (ConnectionString). Tento řetězec jsem uložil do statické proměnné typu string k jejímu pozdějšímu využití. Příklad připojovacího řetězce uvádím níže.

```
Data Source=PC\SQLEXPRESS; Initial Catalog = NemocniceOstrava; Integrated Security=True,
```

Pro všechny proměnné, které jsme v aplikaci používali na více místech, jsme pro snazší správu v případě jejich změny vytvořili speciální třídu StaticAtributes.cs. Zde se ukládaly proměnné a metody, ke kterým bylo potřeba mít okamžitý přístup odkudkoliv z aplikace, bez nutnosti vytvoření instance této třídy.

Vložením připojovacího řetězce do kontruktoru třídy SqlConnection jsem získal instanci třídy, díky níž je možné použít metodu Open vytvářející připojení k databázi. Nyní jsem mohl začít posílat na databázi dotazy. Posílání jsem prováděl pomocí objektu třídy SqlDataAdapter, které jsem jako parametr konstruktoru předal řetězec, představující dotaz a referenci na vytvořené připojení. Vzhledem k tomu, že se jedná o desktopovou aplikaci kdy k ní přistupuje vždy jen jeden uživatel na jednom počítači, tak jsem neřešil problém transakcí. Uvádím příklad dotazu zasílaného na databázi, získávajícího všechny konfigurační nastavení.

```
SqlDataAdapter a = new SqlDataAdapter("SELECT * FROM [NemocniceOstrava].[dbo].[nastaveni]", connection);
```

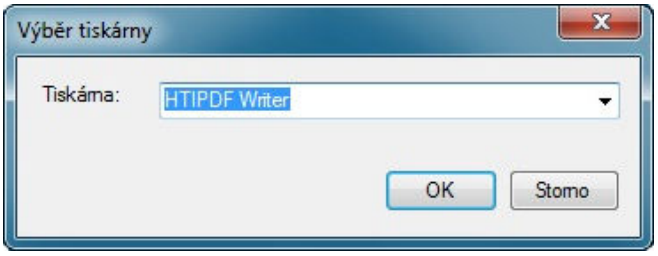
Další postup se pak liší dle typu použitého dotazu. Například při použití dotazu typu SELECT jsem využil podtřídu třídy SqlDataAdapter SelectCommand, a její metodu ExecuteReader. Její pomocí jsem získal instanci třídy SqlDataReader, která do sebe ukládá všechny řádky databáze odpovídající zadanému dotazu. Pro získání dat z tohoto objektu jsem použil metodu Read, která umístěná například do cyklu while postupně prochází všechny uložené záznamy. K jednotlivým hodnotám se přistupuje podobně jako k prvkům v poli nebo v kolekci.

```
SqlDataReader sdr = a.SelectCommand.ExecuteReader();
string s = sdr[1].ToString();
```

Číslo v závorkách označuje sloupec databáze jehož data chceme získat. Důležitým faktorem je, že první sloupec v tabulce má číslo 1, a ne jako u prvků pole 0. Celý tento podrobný popis připojení a práce s SQL Serverem pomocí programovacího jazyka C# je základem pro napojení ovládacích prvků aplikace jako jsou TextBoxy, SelectListy, Checkboxy a další.

5.2 Tisk jednotlivých záznamů pacientů

Před samotný řešením jsem si rozdělil tento problém do dvou bodů. Prvním bylo vyhledání všech nainstalovaných tiskáren a výběr některé z nich k tisku. Druhým vytvoření a samotný tisk dokumentu. Ujasnili jsme si s druhým členem týmu odkud a jak se bude metoda tisku volat. Domluvili jsme se, že se tisk bude spouštět kliknutím na tlačítko, umístěné pouze na obrazovce s přehledem již vytvořené karty pacienta. Abych nebyl závislý na rozložení prvků na obrazovce s detailem pacienta, rozhodl jsem se pro výběr tiskárny vytvořit vlastní dialogové okno. Protože se jedná jen o okno, ve kterém si uživatel vybere některou z dostupných tiskáren, není nutné mít v něm více ovládacích prvků než tlačítka na potvrzení, nebo zrušení tisku a ComboBox k výběru tiskárny. Výsledné okno je vidět na Obr.3



Obr.2 Vzhled dialogového okna pro výběr tiskárny

K získání názvů nainstalovaných tiskáren jsem použil třídu PrinterSettings. Zde se nachází kolekce InstalledPrinters uchovávající v sobě všechny nainstalované tiskárny na aktuálním použitém zařízení. Příkazem foreach jsem prvky z této kolekce postupně naplnil ComboBox. Tuto operaci jsem prováděl vždy při spouštění dialogového okna, proto je tedy umístěna v konstruktoru dialogového okna. Celá třída reprezentující dialogové okno má v sobě už jen kromě zmíněného konstruktoru pouze atribut, uchovávající v sobě název vybrané tiskárny. Následně podle vybraného tlačítka se určí, zda aplikace přejde k samotnému tisku, nebo ne. Toho bylo docíleno tak, že v podmínce jsem otestoval, zdali dialogové okno vrací DialogResult.OK. V tom případě aplikace nastavila tiskárnu pro tisk v objektu z třídy PrintDocument.

```
PrintDocument printDoc = new PrintDocument();
printDoc.PrinterSettings.PrinterName = dlg.NazevTiskarny;
```

Dalším atributem, který se musí před samotným tiskem nastavit je kvalita tisku. Seznam dostupných kvalit poskytnutých tiskárnou se nachází opět ve třídě PrinterSettings, ale tentokrát v její konkrétní instanci kterou získáme z dříve vytvořeného objektu třídy PrintDocument. Kvalita tisku je reprezentována objektem PrinterResolution. Vzhledem k tomu, že nebylo za úkol poskytnout uživateli možnost výběru kvality tisku, nastavil jsem pro tiskárnu automaticky tisk ve vysoké kvalitě. Kód této operace je uveden níže:

```
foreach (PrinterResolution printRes in PrintDoc.PrinterSettings.PrinterResolutions)
{
        if (printRes.Kind == PrinterResolutionKind.High)
        {
                printDoc.DefaultPageSettings.PrinterResolution = printRes;
        }
}
```

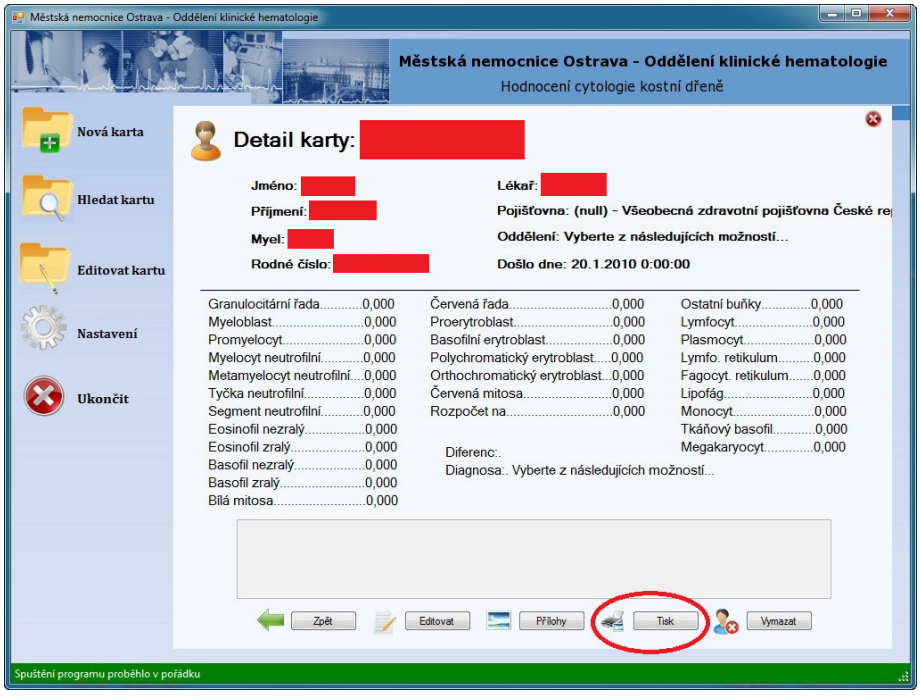
Nyní když je nastavena tiskárna pro tisk i kvalita tisku. Zbývá z informací o pacientovi vytvořit samotný dokument pro tisk. Průběh je podobný jako při vykreslování obrázku na plátno. Vytvořil jsem nový objekt typu Graphics pomocí argumentu třídy PrintPageEventArgs. A pomocí metody DrawString začal na něj vypisovat (vykreslovat) jednotlivé informace. Metoda DrawString má 5 přetížení. Pro svůj účel jsem vybral její podobu ve tvaru:

```
DrawString(string s, Font f, Brush brush, Point point)
```

Text uvedený v dokumentu jsem získával z vyplněných ovládacích prvků. Před samotným vypisováním informací, bylo třeba definovat používané typy písma.

```
Font textFont = new Font("Arial", 12);
Font nadpisFont = new Font("Arial", 20);
```

Pro rozdělení jednotlivých částí stránky jsem použil vodorovné čáry, vykreslené metodou DrawLine. Pro ilustraci slouží obrázek Obr.3 a příloha B. Na obrázku je vidět aplikaci zobrazující detail záznamu o pacientovi a v příloze se nachází vzor vytištěné stránky.



Obr.3 Vzhled aplikace při zobrazení detailu o pacientovi

5.3 Přidávání obrázkových příloh k jednotlivým záznamům pacientů

Úplně novou funkcí, kterou předešlá aplikace neumožňovala je přidávání obrázkových příloh k jednotlivým záznamům pacientů. Začal jsem slovním návrhem její implementace.

Přílohy bude možno přidávat po kliknutí na tlačítko umístěné v obrazovce zobrazující detaily o pacientovi. Protože se jedná o obrázkové přílohy, budou zobrazeny na úvodní straně jejich miniatury, které lze po kliknutí zvětšit do skutečné velikosti. Samotné přidávání bude mít několik kroků. Po kliknutí na miniaturu obrázku, proběhne vyhledání nahrávaného snímku v souborovém systému počítače a následně jeho překopírování z původního umístění pod jedinečným názvem do složky určené databází. Samotná cesta k nakopírovanému snímku se nakonec uloží do databáze ke konkrétnímu záznamu pacienta.

Vše tedy začíná vybráním pacienta. Jakmile se zobrazí obrazovka s jeho detaily, uloží se do globální proměnné id_karty, hodnota jeho id z databáze. Po kliknutí na tlačítko příloh se provede přepnutí obrazovky přehledu pacienta na obrazovku s přidáváním příloh. Před jejím vykreslením se projdou v databázi sloupce pojmenované priloha1 až priloha5 a obsahují-li jinou hodnotu než NULL nebo prázdný řetězec, vykreslí se do příslušných ovládacích prvků miniatury daných snímků. Samotné vykreslení miniatury probíhá v následující části kódu.

```
path2 = myReader["priloha2"].ToString();
if (!path2.Equals("") && !path2 != null)
{
    pictureBox15.Image = Image.FromFile(path2);
}
else
{
   pictureBox15.Image = null;
}
```

Jako další bylo potřeba implementovat metodu pro nahrávání příloh. Po stisku na miniaturu se uživateli zobrazí dialogové okno k vybrání souboru. Aby se uživateli nezobrazovaly ve složkách všechny uložené soubory, aplikoval jsem na toto dialogové okno souborový filtr, díky němuž se uživateli zobrazují jen soubory s příponou jpg, popřípadě jpeg. Aplikování filtru uvádí kód níže.

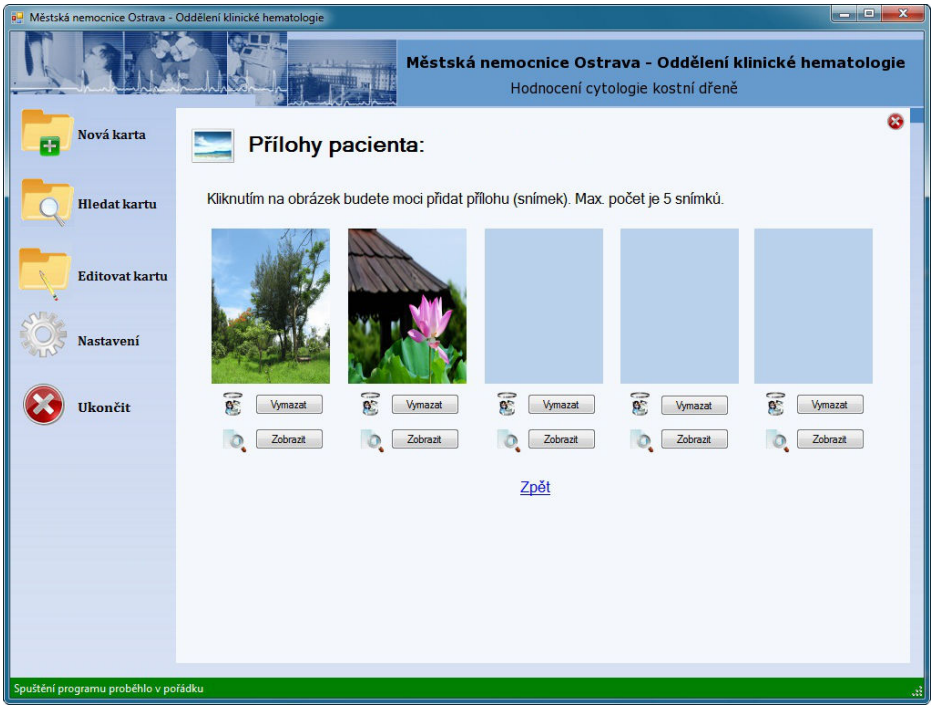
```
OpenFileDialog open = new OpenFileDialog();
open.Filter = "Image Files(*.jpg; *.jpeg)|*.jpg; *.jpeg";
```

Po vybrání souboru se cesta k němu získá pomocí atributu Filename z instance objektu dialogového okna. Následuje překopírování vybraného souboru pomocí metody Copy třídy File.
Tato metoda obsahuje dvě přetížení. Použil jsem to, u kterého lze specifikovat i vlastnost, má-li se v případě, že v cílové složce už soubor s daným názvem existuje přepsat tento soubor novým, který je právě nahráván. Použitím toho jsem zároveň vyřešil i problém s editací dané přílohy, kdy ani není vyžadováno, aby byla původní příloha nejdříve smazána. Název pod který se soubor ukládá je složen názvu snimek, identifikátoru vybraného pacienta a číslem nahrávané přílohy. Právě díky identifikátoru pacienta, který je vždy jedinečný, mám jistotu, že se nestane**,** aby přílohy jednoho pacienta se přepsaly přílohy jiného pacienta. Po nakopírování se provede ještě úprava cesty k příloze v databázi a zobrazí se nám miniatura obrázku.

Vymazání přílohy se provádí pomocí tlačítka vymazat pod každou přílohou. Samotné smazání se skládá zaprvé z uvolnění objektu obrázku v miniatuře a zadruhé z vyhledání fyzického umístění souboru v úložišti pomocí cesty zapsané v databázi a jeho vymazání metodou Delete třídy File. Bez uvolnění objektu v miniatuře by nešel soubor přílohy smazat z důvodu využívání jeho instance v miniatuře. Před samotným smazáním jsem ještě raději nechal aplikaci zkontrolovat existenci

mazaného souboru, aby nedošlo k vyvolání výjimky. Nakonec jsem v databázi nastavil atribut představující cestu k dané příloze na NULL.

Pro zobrazení přílohy v plné velikosti jsem vytvořil vlastní jednoduché dialogové okno, které kromě tlačítek pro minimalizaci, maximalizaci a zavření obsahuje pouze ovládací prvek, do kterého se soubor přílohy zobrazuje v plné velikosti. Je-li snímek větší než umožňují rozměry dialogového okna, lze k navigaci využít posuvníků na boku popřípadě naspodu okna. Tímto poměrně jednoduchým způsobem jsem obsáhl veškerou funkčnost správy obrázkových příloh. Na Obr.4 je vidět obrazovka aplikace s přidáváním příloh.



Obr.4 Vzhled obrazovky s přidáváním příloh k pacientům

6. Vytvoření instalačního balíčku a zprovoznění aplikace v nemocnici

Po vytvoření konečné verze naší aplikace obsahující všechny části bylo potřeba ji zprovoznit také na cílovém počítači v nemocnici.

K tomuto účelu jsme ve Visual Studiu vytvořili samoinstalační balíček, který je dokonce i schopen v případě, že na počítači chybí některá ze součástí potřebná ke spuštění aplikace, tuto součást identifikovat a vyhledat na internetu. Dále jsme ke zprovoznění připravili CREATE skripty, pro rychlé vytvoření databázové struktury na serveru v nemocnici. Tyto skripty jsme vygenerovali programem Oracle SQL Developer Data Modeler, který umožňuje vygenerování struktury navržené databáze ve formě SQL příkazů přímo pro vybraný typ databázového serveru. S vytvořením databáze nebyl problém, protože v nemocnici jsme měli k dispozici SQL Server Management Studio.

Větší problém byl se zprovozněním samotné aplikace. Samoinstalační balíček nám neobjevil žádnou chybějící součást, která by byla vyžadována ke spuštění aplikace, ale přesto byla vyvolána výjimka oznamující chybějící součást. Nepomohlo přeinstalování aplikace, ani doinstalování vyžadované součásti ručně. Pomohla nám testovací verze Visual Studia, ze které vytvořený samoinstalační balíček fungoval bez problémů. Nakonec jsme ještě nastavili připojovací řetězec k databázovému serveru k nemocnici a tím bylo zprovoznění aplikace hotovo.

7. Teoretické a praktické znalosti získané v průběhu studia uplatněné v průběhu praxe

V průběhu bakalářské praxe jsem velmi uplatnil své znalosti získané v předmětech Úvod do programování v jazyce C#, Úvod do softwarového inženýrství, Teorie zpracování dat, Tvorba informačních systémů a Správa databází. Každý z těchto předmětů mi poskytl solidní základ, na kterém jsem mohl ověřit své znalosti a dovednosti potřebné pro tvorbu aplikace.

Obzvláště předmět Tvorba informačních systémů, který jsem absolvoval ve stejné době jako bakalářskou praxi, mi svou náplní velice pomohl při vytváření některých částí aplikace.

8. Znalosti a dovednosti scházející v průběhu odborné praxe

Jediné co mi v průběhu praxe z dovedností chybělo, byla schopnost tvorby komplikovanějších SQL příkazů. Tento nedostatek jsem však během praxe samostudiem vyřešil.

9. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Během praxe jsem se velmi zlepšil v programovacím jazyce C#. Navíc jsem měl možnost podílet se na procesu vytvoření aplikace, od jejího prvotního návrhu, přes implementaci, až po konečné nasazení na cílové stanici. Další důležitou zkušeností, kterou jsem si v praxi vyzkoušel, byla práce v týmu. Tato zkušenost je velmi ceněná hlavně v zaměstnání, kde jen málokdy bývá celá aplikace vytvářena jedním člověkem, a proto se programátor musí naučit psát kód tak, aby mu nerozuměl jen on sám, ale i ostatní programátoři.

Závěrem bych chtěl dodat, že největším potěšením pro mě bylo zjištění, že naše aplikace je již aktivně využívána na oddělení klinické hematologie a že máme již první odezvy a připomínky od jejich uživatelů, čímž naše spolupráce s Městskou nemocnicí v Ostravě ještě nekončí.